Implementing Sequential Prefixspan Algorithm by Using Static Load Balancing

[1]Kokkonda Spandana [2] Mr. N Naveen Kumar

[1]M.Tech, CSE Department, JNTUH School Of Information Technology, Village Kukatpally, Mandal Balanagar, Dist Medchal, Telangana, India.

[2]Assistant Professor, Cse Department, JNTUH School Of Information Technology, Village Kukatpally, Mandal Balanagar, Dist Medchal, Telangana, India.

Abstract: Repeated series mining is well known and well studied trouble in information mining. The productivity of the formula is used in several alternative regions like chemistry, bioinformatics, and market basket analysis. A completely unique parallel algorithmic rule for mining of frequent sequences supported a static load-balancing is planned. The static load balancing is done by measure the machine time using a probabilistic algorithm. For cheap size of instance, the algorithms deliver the good speedups. The conferred approach is extremely universal: it is often used for static load-balancing of alternative pattern mining algorithms like item set/tree/graph mining algorithms.

*****

1. INTRODUCTION

Incessant example mining is an important data mining strategy with a large assortment of mined examples. The mined incessant examples are sets of things (item sets), successions, charts, trees, and so on. Regular grouping mining was at the start represented. The GSP calculation introduced within the initial to tackle the problem of normal grouping mining. Because the continuous arrangement mining is an augmentation of item set mining, the GSP calculation is an augmentation of the A priori calculation. The A priori and therefore the GSP calculations are expansiveness initial pursuit calculations. The GSP calculation endures with comparative problems because the A priori calculation: it's moderate and memory expenditure. As an outcome of the gradualness and memory utilization of calculations represented, totally different calculations were planned. The 2 noteworthy thoughts within the regular succession mining are those of Zaki and I. M. Pei and Han dynasty. These 2 calculations utilize the supposed prefix based sameness categories (PBECs in short), i.e., speak to the instance as a string and parcel the arrangement of all examples into disjoint sets utilizing prefixes. The 2 calculations vary simply within the knowledge structures won't to management the inquiry. The algorithms represented are fast. In any case, at the purpose once the consecutive calculation keeps running for an extremely long term there's a demand for parallel calculations. For instance, the one portrayed during this paper, there's a very regular likelihood to position a subjective continuous grouping mining calculation: section the arrangement of each single regular succession utilizing the PBECs. The PBECs are created, planned, and executed on the processors. Since the PBECs are planned once, static burden parity of the calculation is processed. This technique has one most well-liked standpoint: it counteracts rehashed colossal exchanges of data among hubs (the information is changed once among processors); what is a lot of, one hindrance: assessing the live of a PBEC may be a computationally troublesome issue. As of now, there do not m exist versatile parallelization's of those calculations. There are 2 varieties of parallel PCs: shared memory machines and disseminated memory machines. Parallelizing on the mutual memory machines is a smaller amount difficult than parallelizing on disseminated memory machines. The dynamic burden adjusting is straightforward on shared memory machines, because the instrumentation bolsters easy parallelization: the processors have entry to the whole info. For this work, disseminated memory machines, i.e., bunch of workstations, was utilized. Inspecting system that statically stack alter the calculation of parallel regular item set mining procedure, are planned; In these 3 papers, the supposed twofold testing procedure and its 3 variations were planned. This work amplifies the thought exhibited to parallel continuous grouping mining calculation. The twofold inspecting procedure is improved by presenting weights that speaks to the relative making ready time of the calculation for a particular PBEC.

2. RELATED WORK

In the Load equalization necessary things are estimation of load, comparison of load, stability of various system, performance of system, interaction between the information sets, nature of labor to be transferred, choosing of information sets and lots of alternative ones to think about whereas developing such algorithm Sampling technique that statically load-balance the computation of parallel frequent item set mining method, are projected within the double sampling method is increased by introducing weights that represents the relative time interval of the algorithmic rule for a specific PBEC. Alternative algorithms were projected.

The 2 major ideas within the frequent sequence mining are those of Zaki and architect and Han. These 2 algorithms use the alleged prefix primarily based equivalence categories (PBEC sin short), i.e., represent the pattern and partition the set of all patterns into disjoint sets exploitation prefixes. The 2 algorithms take issue only within the knowledge structures won't to management the search. The sequent algorithmic rule runs for too long there's a necessity for parallel algorithms. Like the one delineate during this paper. There's a really natural chance to lay an arbitrary frequent sequence mining algorithm: partition the set to fall frequent sequences exploitation the PBECs. The GSP algorithm given in is that the initial to resolve the matter of frequent sequence mining. Because the frequent sequence mining is an extension of item set mining, the GSP algorithmic rule is an extension of the A priori algorithm. The A priori and also the GSP algorithms are breadth initial search algorithms. The GSP algorithm suffers with similar issues because the A priori algorithm: it's slow and memory consuming. Free span algorithm is an example of 1 of the primary DFS algorithms. The algorithm was increased within the Prefix- span algorithm that uses the pseudo projected information format, introduced for frequent item set mining. The pseudo-projected information is actually terribly the same as the vertical illustration of the information utilized in the Spade algorithm. Our technique uses the Prefix span algorithm and its operations as a base sequent algorithm. There's additionally AN algorithm that extends the tree projection algorithm for mining of frequent things to sequences.

3. FRAME WORK

Proposed could be a novel parallel technique that statically load-balance the computation. That is: the set of all frequent sequences is initial split into PBECs, the relative execution time of every PBEC is calculable and eventually the PBECs are assigned to processors. The strategy estimates the interval of 1 PBEC by the consecutive Prefix span formula exploitation sampling. During this section, we tend to make a case for the intuition behind the method. It's necessary to remember that the period of time of the serial formula scales with: 1) the information size; 2) the amount of frequent sequences; 3) the amount of embeddings of a frequent sequence in information transactions.

3.1 The whole database D is used to run a consecutive formula on the information and sample the output of the formula, i.e., the set of all frequent sequences F. Such approach doesn't create sense: the consecutive formula is dead on the complete information D. Therefore, it runs for a minimum of constant quantity of time because the consecutive formula we tend to use for comparison of the speed of our parallel formula.

3.2 A Database sample Ď⊆D is used to run a sequent formula exploitation the relative support, manufacturing F '. F' is used as an approximation to F, however, F' will be quite vast. Therefore, the sample F's ⊆F' is used for partitioning and planning. Such an algorithmic rule reduces the execution time of the serial formula by reducing the information size: |Ď|<<|D|. For a PBEC [S], the value |[S] ∩F'|/|F'| estimates the relative processing time of a PBEC.|[S] ∩F'|/|F'| is estimated by |[S] ∩F's|/|F's|. We call this approach the double sampling process.

The Prefixspan algorithm is build on the operations described above, see Algorithm 1 and 2. Initial pseudo-projection is performed in Algorithm 1 Collection of frequent extensions is performed in Algorithm 2. The two projection operations are used. Please note that there are two kinds of items of the Algorithm 2. The items that open new event and items that are appended to the last event. From the previous description follows that the overall computational complexity of the algorithm depends solely on the database D and the minimal support value.

Algorithm 1. Prefixspan Sequential

```
PREFIXSPAN-SEQUENTIAL(Database $\mathcal{D}$,
    Integer $min\_supp$)
1: $\Sigma \leftarrow$ all frequent extensions from $\mathcal{D}$
2: for for all $e \in \Sigma$ do
3:     $Q \leftarrow \langle(e)\rangle$
4:     create projection $\mathcal{D}|_Q$
5:     PREFIXSPAN-MAINLOOP($\mathcal{D}, Q, \mathcal{D}|_Q, min\_supp$)
6: end for
```

Algorithm 2. Prefixspan Sequential – Mainloop

```
PREFIXSPAN-MAINLOOP(Database $\mathcal{D}$, Sequence $Q$,
    Database $\mathcal{D}|_Q$, Integer $min\_supp$)
1: Collect all extensions and its support $\Sigma$ of $Q$ using $\mathcal{D}$ and
   $\mathcal{D}|_Q$.
2: for each $X \in \Sigma$ do
3:     $Q' \leftarrow Q$ extended with $X$. {$X$ can be either $e_{(}$ or $e, e \in \mathcal{B}$}
4:     if $\sigma(Q', \mathcal{D}) < min\_supp$ then continue
5:     Create $\mathcal{D}|_{Q'}$ using $\mathcal{D}, \mathcal{D}|_Q$, and $X$.
6:     PREFIXSPAN-MAINLOOP($\mathcal{D}, Q', \mathcal{D}|_{Q'}, min\_supp$)
7: end for
```

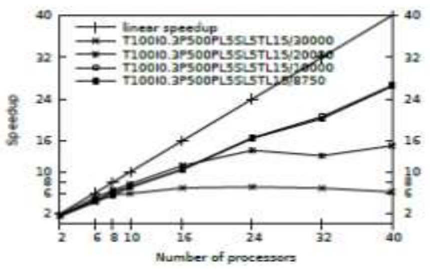
This section contains the main contribution of the paper. All the ideas presented in the previous sections are integrated here, showing how to execute the Prefixspan in parallel. The parallel Prefixspan algorithm has four phases. In the Phase 1, the method produces the weighting tree T containing the estimates of the relative processing time of the PBECs. In the Phase 2, the method partitions the set F into PBECs, using the tree T, and schedule PBECs on processor. In the Phase 3, the method distributes the database in such a way that each processor can process independently its assigned PBECs. In the Phase 4, the method executes the Prefixspan algorithm in parallel on all processors, processing its assigned PBECs. The algorithm is summarized in the motivation behind Algorithm 3 is that the algorithm time increase when: 1) dataset size increase; 2) the support decreases, or in another words when the size of F increase.

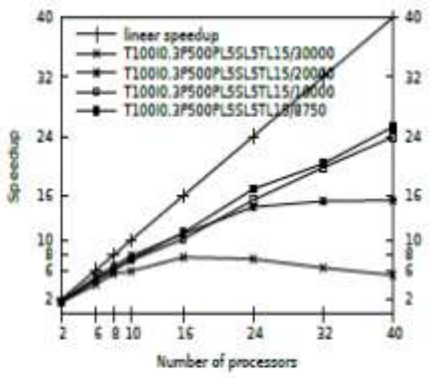
Algorithm 3. PREFIXSPAN-PARALLEL(Database $\mathcal{D}$, Integer *min_supp*)

```
1: Phase 1: Execute an arbitrary frequent sequence miner on
   database sample $\widetilde{\mathcal{D}} \subseteq \mathcal{D}$, create $\mathcal{T}$.
2: Phase 2: Partition $\mathcal{F}$ into PBECs and schedule them on the
   processors.
3: Phase 3: Exchange database among processors.
4: Phase 4: Execute an arbitrary sequence miner in the assigned
   PBECs.
```

4. EXPERIMENTAL RESULTS

In this section, we through an experiment valuate the proposed technique. The entire algorithm was implemented in C++ (compiled with gcc 4.4) exploitation MPI, resulting in _ 30'000 lines of code. The implementation was executed on the CESNET metacentrum on the zegox cluster. Every zegox's node contains 2 Intel E5-2620 equipped with 1_-Infiniband. Nodes were completely allocated for these measurements and used a maximum of five cores per node (to avoid influences from different jobs).





One event was made up of ids of the resources fetched in a very window of 10 seconds by one IP address. From the transactions, items were removed if conferred in each transaction. In Figure four are shown the speedups of our methodology. All of the projected methods have speedups up to 20– 32 on 40 processors for lower values of support. These 3 ways exhibits similar performance on the datasets generated exploitation the IBM generator. The speedups are lower, for higher values of support. for instance, the T1000I0.3P500PL5SL5TL15 dataset has quite good speedups for supports 10'000 an 8'750 and unhealthy speedups for supports 30'000 and 20'000.

5. CONCLUSION

We proposed an algorithmic program for mining of frequent sequences exploitation static load equalization. The strategy creates a sample of frequent sequences and uses this sample for estimating the relative quantity of the rule inside the PBECs. Assess of the relative amount is in reality performed by estimating method quality of process varied PBECs. The relative interval is then used for partitioning and programming of the PBECs. The matter is that the computable size of a PBEC depends on the event of the PBEC. This dependency may be altogether chance removed by exploitation.

References

[1] (2016). Description of the round robin tournament on Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Round_ robin_tournament

[2] R. Agrawal and J. C. Shafer, "Parallel mining of association rules," IEEE Trans. Knowl. Data Eng., vol. 8, no. 6, pp. 962–969, Dec. 1996.

[3] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in Proc. 20th Int. Conf. Very Large Data Bases, 1994, pp. 487–499.

[4] R. Agrawal and R. Srikant, "Mining sequential patterns," in Proc. 11th Int. Conf. Data Eng., 1995, pp. 3–14.

[5] V. Chv atal, "The tail of the hypergeometric distribution," Discr. Math., vol. 25, no. 3, pp. 285–287, 1979.

[6] S. Cong, J. Han, J. Hoeflinger, and D. Padua, "A sampling-based framework for parallel data mining," in Proc.10th ACM SIGPLAN Symp. Principles Practice Parallel Program., 2005, pp. 255–265.

[7] R. L. Graham, "Bounds on multiprocessing timing anomalies," SIAM J. Appl. Math., vol. 17, no. 2, pp. 416–429, 1969.

[8] D. Gunopulos, R. Khardon, and R. S. Sharma, "Discovering all most specific sentences," ACM Trans. Database Syst., vol. 28, pp. 140–174, 2003.

[9] D. Gunopulos, H. Mannila, and S. Saluja, "Discovering all most specific sentences by randomized algorithms," in Proc. 6th Int. Conf. Database Theory, 1997, pp. 215–229.

[10] V. Guralnik, N. Garg, and G. Karypis, "Parallel tree projection algorithm for sequence mining,"in Proc. 7th Int. Euro-Par Conf. Euro-Par Parallel Process., 2001, pp. 310–320.